A7 - Web Architecture Specification

Databases and Web Applications Laboratory (LBAW)
Bachelor in Informatics Engineering and Computation (L.EIC)

Sérgio Nunes
Dept. Informatics Engineering
FEUP · U.Porto

Web Resources

➔ 'Web resources' represent your web application endpoints.

➔ The A7 artefact corresponds to the identification and description of endpoints.

➔ Types of endpoints:

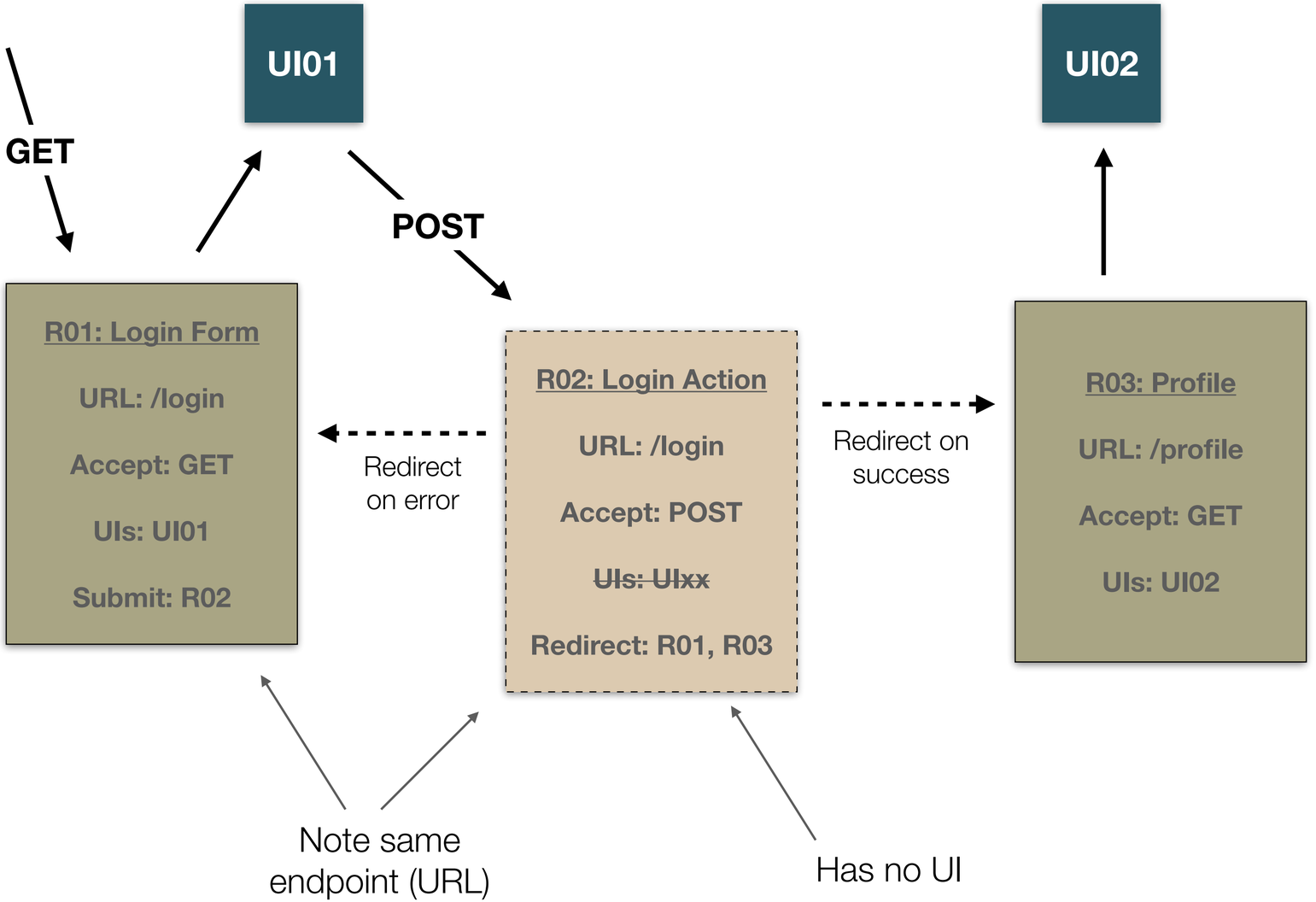
  ➔ **View endpoints**: show user interface to the user (A3 UIs), returns HTML;

  ➔ **Action endpoints**: process information then redirects to view endpoint;

  ➔ **Ajax endpoints**: available for asynchronous calls, returns JSON / HTML*.

➔ ** JSON output requires rendering on the client; HTML is pre-rendered on the server.*

Overall Architecture – "View-Action Pages Pattern"



OpenAPI

➔ Goal: standardize on how APIs are described.

➔ https://spec.openapis.org/oas/v3.1.0

  ➔ "The OpenAPI Specification (OAS) defines a standard, programming language-agnostic interface description for REST APIs, which allows both humans and computers to discover and understand the capabilities of a service without requiring access to source code, additional documentation, or inspection of network traffic."

➔ The OpenAPI Initiative (OAI) is anchored under the Linux Foundation.

➔ An OpenAPI document is a JSON object, and may be represented either in JSON or YAML.

Example OpenAPI Document (YAML)

```yaml
openapi: 3.0.0

info:
  title: Sample API
  description: Optional multiline or single-line description in [CommonMark](http://commonmark.org/help/) or HTML.
  version: 0.1.9

servers:
  - url: http://api.example.com/v1
    description: Optional server description, e.g. Main (production) server

paths:
  /users:
    get:
      summary: Returns a list of users.
      description: Optional extended description in CommonMark or HTML.
      responses:
        '200':    # status code
          description: A JSON array of user names
          content:
            application/json:
              schema:
                type: array
                items:
                  type: string
```

Metadata

```yaml
openapi: '3.0.0'

info:
  version: '1.0'
  title: 'LBAW MediaLibrary Web API'
  description: 'Web Resources Specification (A7) for MediaLibrary'
```

External Documentation

```
externalDocs:
  description: Find more info here.
  url: https://web.fe.up.pt/~ssn/wiki/teach/lbaw/medialib/a07
```

Servers

➔ The servers section specifies the API server and base URL. You can define one or several servers, such as production and sandbox.

➔ All API paths are relative to the server URL.

```
servers:
- url: http://lbaw.fe.up.pt
  description: Production server
```

Tags

➔ Used to grouping API operations (endpoints).

➔ Use for the defined modules.

```
tags:
  - name: 'M01: Authentication and Individual Profile'
  - name: 'M02: Works'
  - name: 'M03: Reviews and Wish list'
  - name: 'M04: Loans'
  - name: 'M05: User Administration and Static pages'
```

Paths

➔ In OpenAPI terms, **paths** are endpoints (resources) that your API exposes, and operations are the HTTP methods used to manipulate these paths, such as GET, POST or DELETE.

```
paths:
  /ping:
    ...
  /users:
    ...
  /users/{id}:
    ...
```

View Page (HTTP GET) OpenAPI

```yaml
/login:
  get:
    operationId: R101
    summary: 'R101: Login Form'
    description: 'Provide login form. Access: PUB'

    tags:
      - 'M01: Authentication and Individual Profile'

    responses:
      '200':
        description: 'Ok. Show log-in UI'
```

Action Page (HTTP POST) – Form Submission OpenAPI (1/3)

➔ Endpoint header.

```
[/login:]

  post:
    operationId: R102
    summary: 'R102: Login Action'
    description: 'Processes the login form submission. Access: PUB'
    tags:
      - 'M01: Authentication and Individual Profile'
```

Action Page (HTTP POST) – Form Submission OpenAPI (2/3)

➔ Request body section defines the accepted parameters.

➔ Note the different types, and the required parameters.

```
[/login:]

  [post:]

    requestBody:
      required: true
      content:
        application/x-www-form-urlencoded:
          schema:
            type: object
            properties:
              email:          # <!--- form field name
                type: string
              password:    # <!--- form field name
                type: string
            required:
              - email
              - password
```

Action Page (HTTP POST) – Form Submission OpenAPI (3/3)

➔ All action pages have redirects.

```
      responses:
        '302':
          description: 'Redirect after processing the login credentials.'
          headers:
            Location:
              schema:
                type: string
              examples:
                302Success:
                  description: 'Successful authentication. Redirect to user profile.'
                  value: '/users/{id}'
                302Error:
                  description: 'Failed authentication. Redirect to login form.'
                  value: '/login'
```

Action Page – Logout

```yaml
/logout:

  post:
    operationId: R103
    summary: 'R103: Logout Action'
    description: 'Logout the current authenticated used. Access: USR, ADM'
    tags:
      - 'M01: Authentication and Individual Profile'

    responses:
      '302':
        description: 'Redirect after processing logout.'
        headers:
          Location:
            schema:
              type: string
            examples:
              302Success:
                description: 'Successful logout. Redirect to login form.'
                value: '/login'
```

View Page – Register Form

```yaml
/register:

  get:
    operationId: R104
    summary: 'R104: Register Form'
    description: 'Provide new user registration form. Access: PUB'
    tags:
      - 'M01: Authentication and Individual Profile'

    responses:
      '200':
        description: 'Ok. Show sign-up form UI'
```

Action Page — Register (1/3)

```
  post:

    operationId: R105
    summary: 'R105: Register Action'
    description: 'Processes the new user registration form submission. Access: PUB'
    tags:
      - 'M01: Authentication and Individual Profile'
```

Action Page — Register (2/3)

```
      requestBody:
        required: true
        content:
          application/x-www-form-urlencoded:
            schema:
              type: object
              properties:
                name:
                  type: string
                password:
                  type: string
                email:
                  type: string
                picture:
                  type: string
                  format: binary
              required:
                - email
                - password
```

Action Page — Register (3/3)

```
    responses:
      '302':
        description: 'Redirect after processing the new user information.'
        headers:
          Location:
            schema:
              type: string
            examples:
              302Success:
                description: 'Successful authentication. Redirect to user profile.'
                value: '/users/{id}'
              302Failure:
                description: 'Failed authentication. Redirect to register form.'
                value: '/register'
```

View Page — View User

```yaml
/users/{id}:
  get:
    operationId: R106
    summary: 'R106: View user profile'
    description: 'Show the individual user profile. Access: USR'
    tags:
      - 'M01: Authentication and Individual Profile'

    parameters:
      - in: path
        name: id
        schema:
          type: integer
        required: true

    responses:
      '200':
        description: 'Ok. Show the view user profile UI'
```

AJAX Page – Search Works (1/3)

```
/api/works:
  get:
    operationId: R202
    summary: 'R202: Search Works API'
    description: 'Searches for works and returns the results as JSON. Access: PUB.'
```

AJAX Page – Search Works (2/3)

```
      parameters:
        - in: query
          name: query
          description: String to use for full-text search
          schema:
            type: string
          required: false
        - in: query
          name: item
          description: Category of the works
          schema:
            type: string
          required: false
        - in: query
          name: loaned
          description: Boolean with the loaned flag value
          schema:
            type: boolean
          required: false
        - in: query
          name: owner
          description: Boolean with the owner flag value
          schema:
            type: boolean
          required: false
        - in: query
          name: classification
          description: Integer corresponding to the work classification
          schema:
            type: integer
          required: false
```

AJAX Page – Search Works (3/3)

```
      responses:
        '200':
          description: Success
          content:
            application/json:
              schema:
                type: array
                items:
                  type: object
                  properties:
                    id:
                      type: string
                    title:
                      type: string
                    obs:
                      type: string
                    year:
                      type: string
                    owner:
                      type: string
                    type:
                      type: string


continue -->
```

```
--> continued

                  example:
                    - id: 1
                      title: Rihanna - Unapologetic
                      obs: Good pop music album.
                      year: 2012
                      owner: Joana Lima
                      type: MP3
                    - id: 15
                      title: Mr. Bean
                      obs: The best comedy movie.
                      year: 1995
                      owner: Manuel Teixeira
                      type: DVD
```

GitLab OpenAPI Rendering

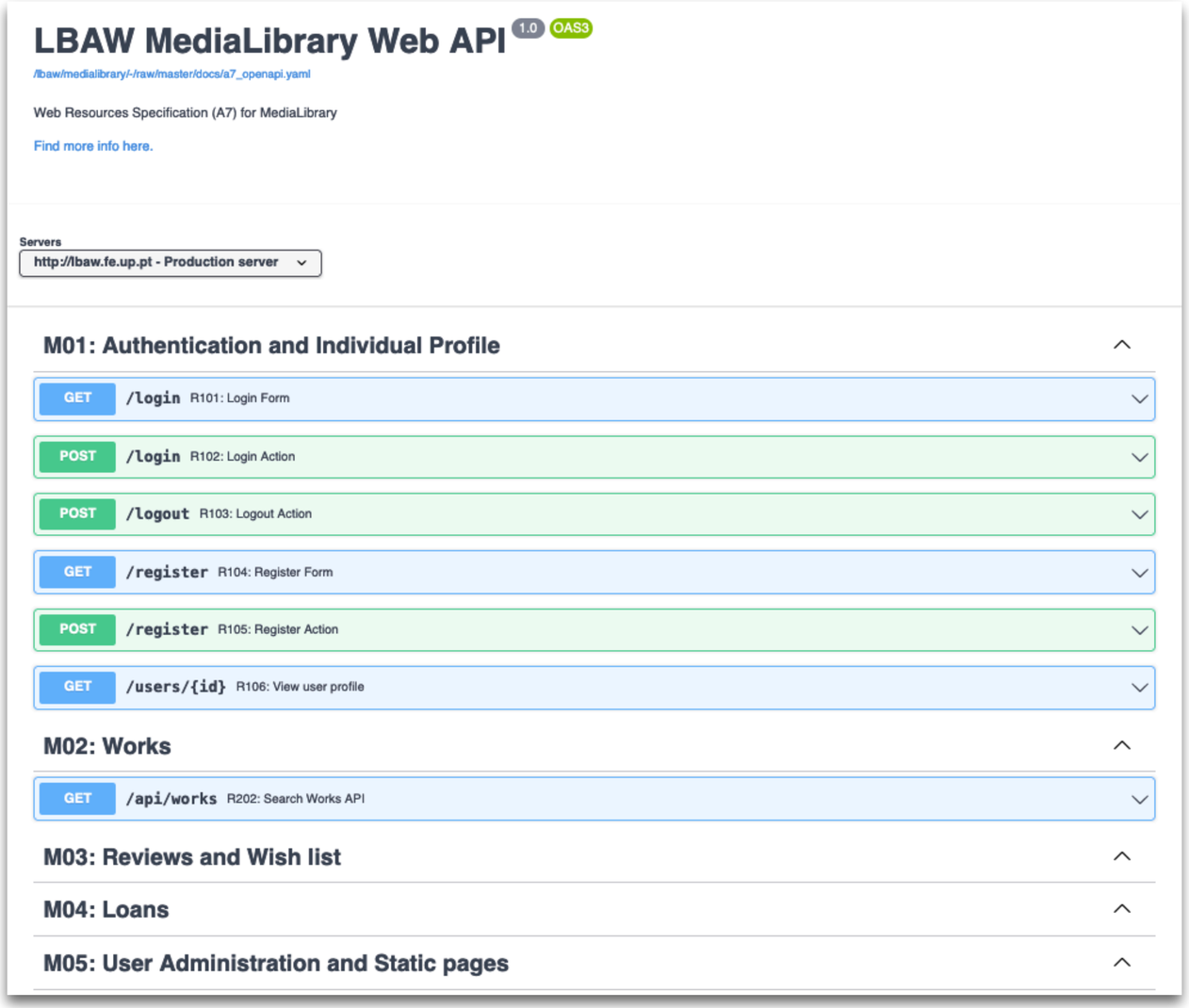
➔ GitLab provides an interactive human-readable view to a OpenAPI specification

➔ File type must be YAML (.yaml) and include 'openapi' in filename

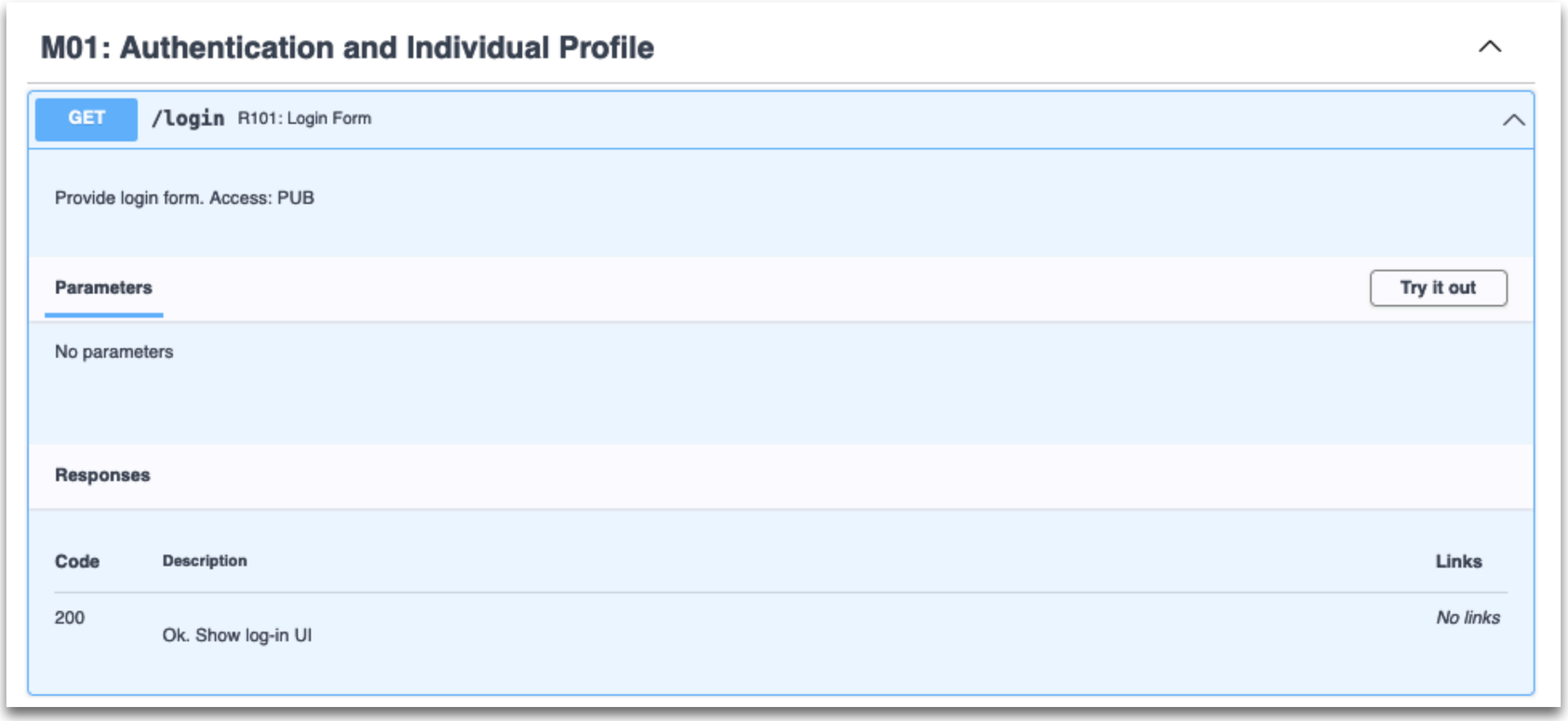
➔ Recommended: a7_openapi.yaml

➔ MediaLibrary: GitLab > lbaw > code-examples > medialibrary > a7_openapi.yaml

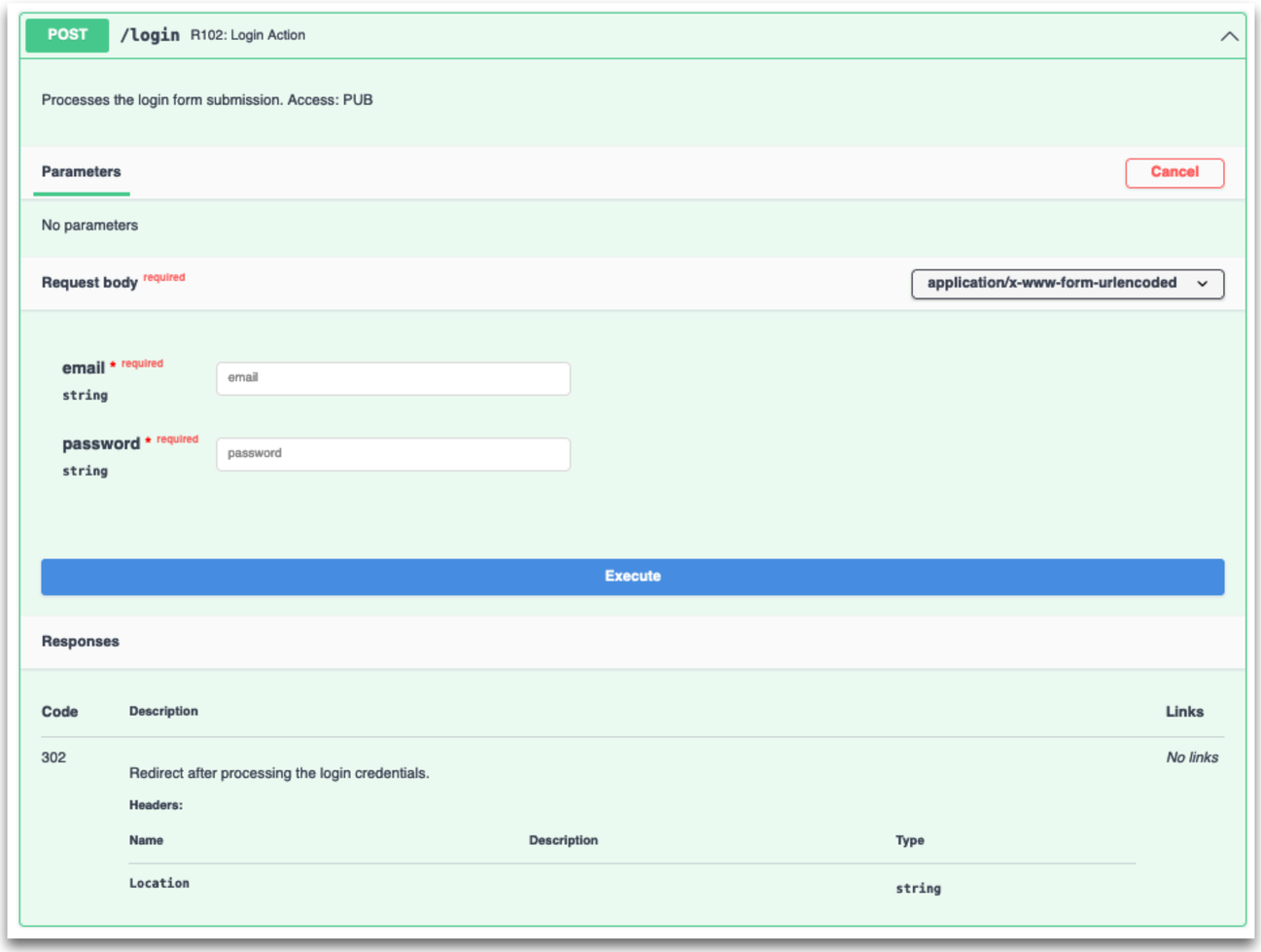
GitLab OpenAPI Rendering



GitLab OpenAPI Rendering



GitLab OpenAPI Rendering



OpenAPI References

- ➔ There is a growing collection of OpenAPI related tools
  - ➔ Data Validators
  - ➔ Documentation
  - ➔ GUI Editors
  - ➔ SDK Generators
  - ➔ Text Editors
  - ➔ ...
- ➔ Check https://openapi.tools